An Investigation of the Analogy between the Random Network and the Hopfield Network

Aristidis Likas and Andreas Stafylopatis
Computer Science Division, Department of Electrical Engineering
National Technical University of Athens, 157 73 Zographou, Athens, Greece

Abstract

In recent publications, a new neural network model, called the *random network*, has been introduced, in which excitation and inhibition are represented by positive and negative signals. It has been shown that, under markovian assumptions, this model has a product form solution, and several issues have been investigated concerning existence and uniqueness of this solution. An analogy between this model and formal neural network models has been established. An extension to the random network model involving positive and negative nodes has also been developed and its ability of acting as autoassociative memory has been examined. In this paper, we study the analogy between the random network with positive and negative nodes and the Hopfield network model and investigate several issues in the context of associative memory operation.

1 Introduction

In recently published work [2, 3, 4], a new neural network model has been introduced. This model is referred to as the *random network model* and is characterized by the existence of *positive* and *negative* signals, which represent excitation and inhibition respectively. Each node accumulates signals and can fire if its total signal count at a given instant of time is strictly positive. When firing, a node sends signals to other nodes or to the outside of the network. Firing occurs at random, the intervals between successive firing instants following an exponential distribution of constant rate. The total signal count at a node at time t is called the *potential* of the node. A negative signal arriving to a node reduces its potential by 1 (it cancels an existing positive signal) or has no effect if the potential is already zero, while an arriving positive signal increases the potential by 1. Signals can either arrive to a node from the outside of the network or from other nodes. External arrivals to each node constitute Poisson streams of positive or negative signals. The movement of signals from node to node in the network is purely markovian following a set of routing probabilities. It is shown in [2] that the above described markovian network has product form solution; that is, the network's stationary probability distribution can be written as the product of the marginal probabilities of the state of each node. Existence and uniqueness of the solution can be established for several classes of network models, such as *feedforward* networks, *balanced* and *damped* networks [2, 3].

In [5, 6], we have proposed an extension to the random network model introduced in [2, 3], which includes positive and negative nodes and preserves the product form solution and the stability properties of the original random network. The network's ability of acting as autoassociative memory has been examined and several techniques have been developed concerning storage and reconstruction of patterns. In this paper we further examine several properties of the random network related to associative memory operation and, in particular, we investigate the analogy between the random network and the Hopfield network models. In the next two Sections we provide a brief description of the random network model with positive and negative nodes and its operation as associative memory.

2 The Model with Positive and Negative Nodes

In associative memory operation, bipolar representation of patterns generally ensures a better behaviour compared to binary. Since the original random network model, in which positive and negative signals are not equivalent, is better adapted to a binary representation of patterns, we have introduced the following extension. We consider two types of nodes in the network—positive and negative— with opposite roles, thus compensating for each other. A positive node behaves exactly as a node in the random network of [2, 3]. A negative node has a completely symmetrical behaviour, namely only negative signals can accumulate at this node, the role of positive signals being purely suppressive. A positive signal arriving to a negative node cancels a negative signal if the node is not empty or has no effect (is lost) if the node is empty. This extension is in fact mathematically equivalent to the random network model described in [2, 3], in which all nodes are assumed positive.

Following the existence of positive and negative connections between nodes, a signal leaving a node will move to another node as a signal of the same or the opposite category. More specifically, a positive signal which leaves positive node i arrives to node j as a positive signal with probability $p^+(i,j)$ and as a negative signal with probability $p^-(i,j)$. Similarly, a negative signal leaving negative node i arrives to node j as a negative signal with probability $p^+(i,j)$ and as a positive signal with probability $p^-(i,j)$. Also, a signal departs from the network upon leaving node i with probability $d(i)$. We shall have $\sum_{j=1}^{n}[p^+(i,j) + p^-(i,j)] + d(i) = 1$, for $i = 1, \ldots, n$, where n is the number of nodes in the network.

External positive and negative signal arrivals to each node i are considered Poisson with rates $\Lambda(i)$ and $\lambda(i)$ respectively. We also assume that, at each node i, the intervals between successive emmissions of signals follow an exponential distribution with mean $1/r(i)$.

Let us consider a random network with n nodes as described above. Let also $\hat{K}(t)$ be the vector of signal potentials at time t, and $\hat{k} = (k_1, \ldots, k_n)$ be a particular value of the vector. The stationary probability distribution of the network's state is given by $p(\hat{k}) = \lim_{t\to\infty} Prob[\hat{K}(t) = \hat{k}]$ whenever this limit exists. Since both types of nodes accumulate signals of the corresponding sign only, an unsigned value is sufficient for describing the potential of a node, once the node sign is given. If we take into account the distinction between positive and negative nodes, the results obtained in [2, 3] are generalized as follows.

If we denote by $\mathcal{P}$ and $\mathcal{N}$ the sets of positive and negative nodes respectively, the flow of signals in the network can be described by the following equations:

$$\gamma^+(i) = \Lambda(i) + \gamma^+_{int}(i) = \Lambda(i) + \sum_{j\in\mathcal{P}} q_j r(j) p^+(j,i) + \sum_{j\in\mathcal{N}} q_j r(j) p^-(j,i) \quad (1)$$

$$\gamma^-(i) = \lambda(i) + \gamma^-_{int}(i) = \lambda(i) + \sum_{j\in\mathcal{P}} q_j r(j) p^-(j,i) + \sum_{j\in\mathcal{N}} q_j r(j) p^+(j,i) \quad (2)$$

where $\gamma^+_{int}(i)$, $\gamma^-_{int}(i)$ represent the internal arrival rates of positive and negative signals respectively, and

$$q_i = \frac{\gamma^+(i)}{r(i) + \gamma^-(i)}, \quad i \in \mathcal{P} \quad (3)$$

$$q_i = \frac{\gamma^-(i)}{r(i) + \gamma^+(i)}, \quad i \in \mathcal{N} \quad (4)$$

It can be shown that, if a unique non-negative solution $\{\gamma^+(i), \gamma^-(i)\}$ exists to the above equations such that $q_i < 1$, then the steady-state network probability distribution has the form:

$$p(\hat{k}) = \prod_{i=1}^{n} [1 - q_i] q_i^{k_i} \quad (5)$$

An analytical solution of the set of equations (1)–(4) is not always feasible. However, there exist classes of networks, for which existence and uniqueness of the solution can be proved [2, 3, 4, 5].

3 Autoassociative Memory Operation

Suppose that we want to store m bipolar patterns $\{\hat{x}^1, \ldots, \hat{x}^m\}$ of the form $\hat{x}^k = (x^k_1, \ldots, x^k_n)$ in a random network with n nodes. In order to determine the characteristics of the network, we first compute the connection weights using some learning rule and subsequently the parameters $p^+(i,j)$, $p^-(i,j)$, $r(i)$, by using the following correspondence which has been developed in [2] and establishes an analogy between usual neural network representation and the random network model. Considering the non-output neuron i, the parameters of the corresponding node i are chosen as follows:

$$\begin{aligned} d(i) &= 0 && (6)\\ r(i)p^+(i,j) &= w_{ij}, \ \text{ if } w_{ij} > 0 && (7)\\ r(i)p^-(i,j) &= |w_{ij}|, \ \text{ if } w_{ij} < 0 && (8) \end{aligned}$$

Summing over all j, the firing rate $r(i)$ of the non-output node i is set

$$r(i) = \sum_j |w_{ij}| \qquad (9)$$

The description of the network is completed only after characterizing its nodes as positive or negative and specifying external arrival rates. This characterization, however, is not fixed for all patterns, but depends on the input pattern that should be reconstructed during operation. If the i-th element of the input pattern is equal to 1, then the i-th node of the network is set positive and the arrival rates of positive and negative signals to this node are $\Lambda(i) \neq 0$ and $\lambda(i) = 0$ respectively. If the i-th element of the input pattern is equal to -1, then the i-th node is set negative with $\Lambda(i) = 0$ and $\lambda(i) \neq 0$.

Thus, there is a distinction between the fixed part of a network (routing probabilities and firing rates) which is set a priori according to some construction (learning) scheme and a variable part (sign of nodes and external signal arrivals) which is dictated by the pattern applied as input to the network. In the following, the term ‘network’ shall be mainly used to denote the fixed part, whereas the variable part will be expressed in terms of the input pattern.

After solving the non-linear system (1)–(4), we obtain the quantities q_i for $i = 1, \ldots, n$. Following the selection of positive and negative nodes, the operation of the network is intended to reconstruct the pattern by yielding low and high q_i values for nodes corresponding to wrong and correct components of the input pattern respectively. Implementation of the above reconstruction procedure shows that the network clearly tends to enhance the distance between q_i values which are supposed to indicate wrong and correct components. However, this differentiation is not always strong enough and can vary following the input pattern, so that there exists no obvious mechanism for determining the number of wrong components. Since this number is generally not known a priori, we need a means of estimating the quality of the reconstruction, which is equivalent to being in a position to identify a stored pattern.

The idea is to take advantage of some characteristic of the stored patterns which allows us to recognize whether an output pattern is one of them. Moreover, reliability is enhanced if only the stored patterns exhibit the above characteristic.

In conclusion, in order to implement autoassociative memory using the random network model, two issues should be addressed.

- An algorithm for appropriately storing the desired patterns (learning rule).
- A scheme for correctly identifying stored patterns during recall.

The above two problems can be dealt with in common or separately. According to the choices made, several schemes can be constructed [5, 6], which are based either on properties of the random network model or on known neural network techniques. Two of these schemes, the outer-product scheme and the spectral scheme [9], are related to the Hopfield network model.

According to the *outer-product algorithm* (Hebbian rule) the connection weights are determined as follows:

$$w_{ij} = \sum_{k=1}^{m} x_i^k x_j^k, \quad i=1,\ldots,n,\ j=1,\ldots,n,\ i \neq j \tag{10}$$

$$w_{ii} = 0, \quad i=1,\ldots,n \tag{11}$$

In a Hopfield-type associative neural network, all stored patterns should be stable states of the network. This means that once the network has attained such a state, no transition out of this state can occur according to the operation of the network [7]. In other terms, all stored patterns $\{\hat{x}^1,\ldots,\hat{x}^m\}$ must satisfy the following equation [8]:

$$sign(\sum_{j=1}^{n} x_j^k w_{ji}) = sign(x_i^k), \quad i=1,\ldots,n,\ k=1,\ldots,m \tag{12}$$

In the case where the outer-product scheme is used for constructing the connection matrix W, the above property of the stored patterns holds with high probability, especially when the size of the network is significantly smaller than the number of patterns [1, 8]. In general, we cannot guarantee that the stability property holds for every pattern $\hat{x}^k$, especially when m has a magnitude comparable to n. This property is guaranteed for every pattern in the case where all patterns $\hat{x}^k$ are orthogonal. This fact can be established with high probability when the patterns $\hat{x}^k$ are generated from a sequence of symmetric Bernoulli trials. The capacity of this scheme is $m = O(n/\log n)$ [8].

The stability property defined above is a characteristic of stored patterns which is closely related to the operation of the Hopfield network. In the next Section, we introduce an analogous property in the context of the random network model.

4 Consistent Patterns

Definition 1. We shall say that a node of the random network is *consistent with its sign* if the internal arrival rate of signals it can accept is greater than the internal arrival rate of signals it rejects, namely:

$$\gamma^{+}_{int}(i) \quad > \quad \gamma^{-}_{int}(i), \quad i \in \mathcal{P} \tag{13}$$

$$\gamma^{-}_{int}(i) \quad > \quad \gamma^{+}_{int}(i), \quad i \in \mathcal{N} \tag{14}$$

A random network with positive and negative nodes is called *consistent* if all its nodes are consistent with their sign.

Definition 2. Suppose that a bipolar pattern $\hat{x} = (x_1, \ldots, x_n)$ is presented as input to a given random network with n nodes. We shall say that the pattern $\hat{x}$ constitutes a *consistent pattern* with respect to the random network if it makes the random network consistent, i.e. if it yields

$$x_i(\gamma^{+}_{int}(i) - \gamma^{-}_{int}(i)) > 0, \quad i = 1, \ldots, n \tag{15}$$

The interpretation of the above definition is that when a consistent pattern is presented as input, the resulting network has the property that the sign of each node is consistent with its input rate from the other nodes of the network.

It is important to notice the similarity between the definition of a stable pattern in the context of the Hopfield neural network model and the definition of a consistent pattern in the context of the random network model. In the Hopfield model the asynchronous update algorithm is based on the stability property of stored patterns. The same principle can be applied in order to develop algorithms for the random network model that exploit the consistency property of stored patterns.

Suppose that we want to use the random network model with positive and negative nodes as an associative memory. Using one of the well-known learning algorithms (e.g. Hebbian scheme, spectral scheme) we can compute the connection weights and create a random network such that all the stored patterns are consistent (with high probability).

At this point some comments must be made concerning the ability of the Hebbian rule to make the stored patterns consistent. Using the Hebbian rule we can construct a Hopfield neural network having the stored patterns as stable states. The question is: once a random network is constructed using the Hebbian rule, does the stored patterns constitute consistent patterns of the random network?

A stable pattern $\hat{y}$ in the context of the Hopfield model satisfies the following condition for each i:

$$\sum_j y_j w_{ji} I(y_j w_{ji} > 0) \quad > \quad \sum_j |y_j w_{ji}| I(y_j w_{ji} < 0), \quad \text{if } y_i = 1 \tag{16}$$

$$\sum_j |y_j w_{ji}| I(y_j w_{ji} < 0) \quad > \quad \sum_j y_j w_{ji} I(y_j w_{ji} > 0), \quad \text{if } y_i = -1 \tag{17}$$

A consistent pattern $\hat{y}$ in the context of the random network model satisfies the following condition for each i:

$$\sum_{j \in \mathcal{P}} q_j r(j) p^+(j,i) + \sum_{j \in \mathcal{N}} q_j r(j) p^-(j,i) \quad > \quad \sum_{j \in \mathcal{P}} q_j r(j) p^-(j,i) + \sum_{j \in \mathcal{N}} q_j r(j) p^+(j,i), \quad i \in \mathcal{P} \quad (\text{i.e. } y_i = 1) \tag{18}$$

$$\sum_{j \in \mathcal{P}} q_j r(j) p^-(j,i) + \sum_{j \in \mathcal{N}} q_j r(j) p^+(j,i) \quad > \quad \sum_{j \in \mathcal{P}} q_j r(j) p^+(j,i) + \sum_{j \in \mathcal{N}} q_j r(j) p^-(j,i), \quad i \in \mathcal{N} \quad (\text{i.e. } y_i = -1) \tag{19}$$

By taking into account equations (7)–(8), the above equations can be rewritten as follows:

$$\sum_j q_j y_j w_{ji} I(y_j w_{ji} > 0) \quad > \quad \sum_j q_j |y_j w_{ji}| I(y_j w_{ji} < 0), \quad \text{if } y_i = 1 \tag{20}$$

$$\sum_j q_j |y_j w_{ji}| I(y_j w_{ji} < 0) \quad > \quad \sum_j q_j y_j w_{ji} I(y_j w_{ji} > 0), \quad \text{if } y_i = -1 \tag{21}$$

By examining equations (20)–(21) we can observe that if all the q_j were equal, i.e. $q_j = q$, $j = 1, \ldots, n$, then these equations would be identical to equations (16)–(17). A great number of experiments has shown that, if a consistent pattern is presented as input to the random network, the resulting q_i values are close to each other. This is the reason why we consider that the Hebbian rule is good enough for contructing a random network in which the stored patterns would be consistent.

Once a noisy version of a stored pattern is presented as input to the network, the recall procedure can be described as follows. First we solve the non-linear system of equations which describe the steady-state behaviour of the random network model, and compute the analog vector $\hat{q} = (q_1, \ldots, q_n)$. Then we recognize the erroneous nodes using the notion of consistency. We calculate the quantities $\gamma^+_{int}(i)$, $\gamma^-_{int}(i)$, for each $i = 1, \ldots, n$ and check whether the conditions (13)–(14) hold for each i. In case a node i does not satisfy the above condition, there exists an error in the component i of the input pattern. Thus, we can identify erroneous nodes and reconstruct the noisy input pattern.

Identification based on the consistency property provides a simple and efficient technique, which has been tested in many experiments and has shown very good reconstruction capabilities.

In what follows, we make an attempt to incorporate identification into the solution of the non-linear system of equations, thus merging the two phases of autoassociative memory operation, and to suitably use the consistency property of stored patterns in order to obtain a model of operation analogous to that of the Hopfield network.

5 An Asynchronous Solution Method

An asynchronous iterative algorithm that can be used to solve the random network (i.e. to solve the non-linear system of equations) has as follows:

1. Choose an arbitrary initial value q_i $(0 < q_i < 1)$ for each node i, i.e. initialize the solution vector $\hat{q} = (q_1, \ldots, q_n)$.

2. Select at random a node i and calculate its internal input rates $\gamma^+_{int}(i)$, $\gamma^-_{int}(i)$, using the current value of $\hat{q}$. Then, according to equations (3)–(4), compute the new value of q_i.

3. Repeat step 2. until final convergence to a fixed value of the vector $\hat{q}$ (with a predefined precision).

The above algorithm is totally asynchronous and yields the solution vector $\hat{q}$ of the random network if the network is stable, i.e. it satisfies the stability condition $0 < q_i < 1$, $i = 1, \ldots, n$ [3]. If the network is not stable, the algorithm may not converge, or it may converge to q_i values which are greater than 1, which is a fact with no physical interpretation in the context of the random network model.

A parameter that can be used to adjust the q_i values is the external signal arrival rate ($\Lambda(i)$ in the case of positive nodes and $\lambda(i)$ in the case of negative nodes). A large value of this rate can lead a node to saturation, while a small one is preferable in case we desire stable networks. As will be shown later, we can exploit this parameter by assigning an appropriate value to it.

The above described method can be used to solve a random network corresponding to a sign vector $\hat{x} = (x_1, \ldots, x_n)$, where x_i is the sign of node i. We can further extend the method in such a way that it could operate in a manner similar to the Hopfield model.

Our objective is to develop an update scheme that performs two operations simultaneously. More specifically, when a pattern is presented as input to the network, the algorithm should discover the nodes whose sign must change in order for the network to be consistent, while at the same time should compute the q_i values corresponding to the resulting consistent network.

The algorithm has as follows:

1. Choose an arbitrary initial value q_i $(0 < q_i < 1)$ for each node i in the network.

2. Select at random a node i and compute the quantities $\gamma^+_{int}(i)$, $\gamma^-_{int}(i)$. Suppose $i \in \mathcal{P}$. If $\gamma^+_{int}(i) > \gamma^-_{int}(i)$, then compute the new value $q_i = [\gamma^+_{int}(i) + \Lambda(i)]/[(r(i) + \gamma^-_{int}(i)]$. If $\gamma^+_{int}(i) < \gamma^-_{int}(i)$, then node i is inconsistent with its sign. Thus node i becomes negative and its new q_i value is $q_i = [\gamma^+_{int}(i) + \Lambda(i)]/[(r(i) + \gamma^-_{int}(i)]$. If the randomly selected node i is negative, the opposite operations are performed.

3. Repeat step 2. until final convergence to a fixed value of the vector $\hat{q}$ (with some predefined precision).

Using the above asynchronous iterative procedure we start from an inconsistent pattern and proceed trying to remove inconsistency. Finally, we attain a consistent pattern while at the same time we compute the vector $\hat{q}$ corresponding to that pattern.

We can exploit this algorithm in order to perform associative memory operations with the random network model. Using a training algorithm (for example the Hebbian rule), we can assign the consistency property to the patterns we want to store. If an erroneous version of one of the stored patterns is presented as input to the network, the above iterative procedure will change the sign of the nodes that correspond to erroneous components of the input pattern, thus reconstructing the stored (bipolar) pattern. It will also provide the corresponding $\hat{q}$ vector, which represents the analog realization of the pattern on the network. This is a new feature that is not found in previous associative recall algorithms.

The method has been tested in various cases and a very promising behaviour can be reported. It should be noted that the algorithm has exhibited a fast convergence during experiments, although this should be the object of further theoretical investigation.

6 Techniques Based on Saturated Nodes

An interesting situation arises in case we make the following choice for the external signal arrival rates:

$$\Lambda(i) = r(i), \quad i \in \mathcal{P} \tag{22}$$

$$\lambda(i) = r(i), \quad i \in \mathcal{N} \tag{23}$$

In this case, according to equations (3)–(4), we observe that a consistent node i is also saturated (i.e. $q_i > 1$). Thus, we have a scheme where consistent nodes correspond to saturated nodes, while inconsistent nodes remain unsaturated. As a consequence, when a consitent pattern is presented as input to the network, all its nodes become saturated. In the definition of a random network containing saturated nodes [2, 3] it is stated that we can assign the value $q_i = 1$ to a saturated node and exclude that node from the product-form solution.

Thus, under the assumptions (22)–(23), our scheme can be restated as follows:

1. Give the initial value $q_i = 1$ for each $i = 1, \ldots, n$.

2. Select at random a node i and compute the quantities $\gamma^+_{int}(i)$, $\gamma^-_{int}(i)$. Suppose $i \in \mathcal{P}$. If $\gamma^+_{int}(i) > \gamma^-_{int}(i)$, then node i is saturated and the value $q_i = 1$ is assigned. If $\gamma^+_{int}(i) < \gamma^-_{int}(i)$, then node i is inconsistent with its sign and its sign must change. Thus node i becomes negative and its new q_i value is $q_i = [\gamma^+_{int}(i) + \lambda(i)]/[(r(i) + \gamma^-_{int}(i)]$. Since $\lambda(i) = r(i)$ and $\gamma^+_{int}(i) < \gamma^-_{int}(i)$, node i is also saturated and the value $q_i = 1$ is assigned. If the randomly selected node i is negative, the opposite operations are performed.

3. Repeat step 2. until there are no more sign changes in the network.

It is important to notice that, when using the above algorithm, there is no change in the q_i value of nodes, since this value always remains equal to 1. The most important is that this update scheme is exactly the same as the Hopfield update scheme. This can be easily verified by observing the similarity of equations (16)–(17) and (20)–(21) in the case where $q_i = 1$ for each i. So, one could say that the Hopfield update scheme is derived as a special case of the general scheme proposed earlier in this section.

Another update scheme that has been developed is based on the same assumption for external arrival rates, i.e. equations (22)–(23). The difference lies in the property that is related to the patterns we want to store. While in the previous case the stored patterns corresponded to a completely saturated network (and all other patterns leave some nodes unsaturated), in this case the stored paterns have the property of keeping the network stable, i.e. $0 < q_i < 1$ for each $i = 1, \ldots, n$ (and all other patterns lead some nodes to saturation). Due to this property we can refer to the stored patterns as stable patterns (in the random network sense).

A method for constructing a random network that is characterized by the above property uses a reversed version of the Hebbian rule. This means that each connection weight takes a value opposite to the one computed using the Hebbian rule. Due to this fact, when one of the stored patterns is presented as input to the network, it makes all its nodes inconsistent, i.e. the conditions (13)–(14) are not satisfied. Since equations (22)–(23) hold, we can infer that in the case of a stored pattern we have $q_i < 1$ for each $i = 1, \ldots, n$.

If an erroneous pattern is presented as input to the random network, the nodes corresponding to the erroneous components will be saturated. So the application of this update scheme leads to changes in the sign of the saturated nodes, and also yields the q_i values with $0 < q_i < 1$ that correspond to the reconstructed pattern.

References

[1] Aiyer, S.V.B., Niranjan, M. and Fallside, F., *A Theoretical Investigation into the Performance of the Hopfield Model*, IEEE Tr. on Neural Networks, Vol. 1, No. 2, June 1990.

[2] Gelenbe, E., *Random Neural Networks with Negative and Positive Signals and Product Form Solution*, Neural Computation, Vol. 1, No. 4, pp. 502-510, 1989.

[3] Gelenbe, E., *Stability of the Random Neural Network Model*, Neural Computation, Vol. 2, No. 2, pp. 239-247, 1990.

[4] Gelenbe, E., *Theory of the Random Neural Network Model*, in Neural Networks: Advances and Applications, E. Gelenbe (Editor), North-Holland, 1991.

[5] Gelenbe, E., Stafylopatis, A. and Likas, A., *Associative Memory Operation of the Random Network Model*, ICANN-91, Helsinki, June 1991.

[6] Gelenbe, E., Stafylopatis, A. and Likas, A., *Performance of the Random Network Model in Associative Memory Operation*, EHEI Research Report, 1991.

[7] Hopfield, J., *Neural Networks and Physical Systems with Emergent Collective Computational Abilities*, Proc. of the Nat. Academy of Sciences USA, vol.79, pp.2554-2558, 1982.

[8] McEliece, R.J., Posner, E. C., Rodemich, E.R. and Venkatesh, S.S., *The Capacity of the Hopfield Associative Memory*, IEEE Tr. on Inf. Theory, Vol. 33, No. 4, July 1987.

[9] Venkatesh, S.S. and Psaltis, D., *Linear and Logarithmic Capacities in Associative Neural Networks*, IEEE Trans. on Information Theory, Vol. 35, No. 3, May 1989.